

**KEYPORT 717™**

**PROGRAMMERS  
REFERENCE  
MANUAL**

**FOR**

**APPLE® II, II+, IIe**





This manual is for the programmer who wants to design a keyboard for a new or existing program. It is *not* needed to operate the KEYPORT with packaged applications like BASIC, THE FARM, VisiCalc, etc. except to redefine the keys. Use the procedure **Defining A Board** to redefine keys.

### COPYRIGHT

Copyright 1983 by Polytel Computer Products Corp., Tulsa, Oklahoma. All rights reserved. Under the copyright laws, this manual and its accompanying software packages may *not* be copied in whole or in part without the written consent of Polytel Computer Products Corp.

### PATENTS

The KEYPORT 717 is the subject of applications for Letter Patents and Design Patents. Unauthorized duplication of the technology may result in patent infringement.

### CUSTOMER SATISFACTION

The KEYPORT 717 carries a 90-day warranty against defects in material or workmanship, effective from the date of purchase. No other warranties are expressed or implied. In the event of KEYPORT 717 failure during the warranty period, simply return your KEYPORT 717, with the sales receipt, to Polytel Computer Products Corp., 2121 South Columbia, Tulsa, Oklahoma, 74114. If you have any comments or questions about the KEYPORT call 800-331-4411 (in Oklahoma 918-744-9844) between 8:00 a.m. and 5:00 p.m. Central Standard Time. We will welcome your suggestions, so please feel free to call.

### LIMITATION ON WARRANTIES AND LIABILITY

Even though Polytel Computer Products Corp. has tested this product and has reviewed this manual's contents, neither Polytel Computer Products Corp. nor its suppliers make any warranty, expressed, implied or statutory with respect to this manual, the product described in this manual or its related software.

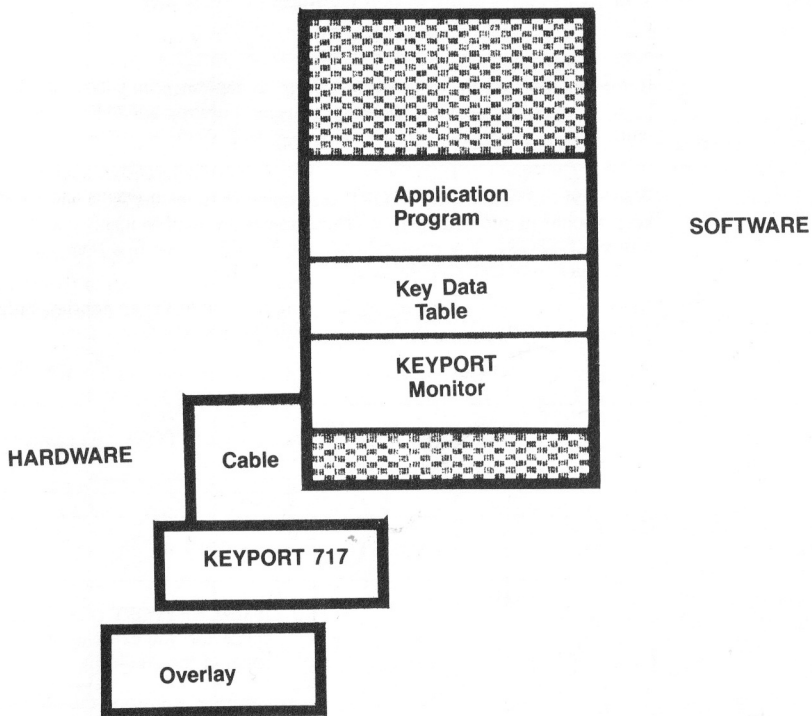


Contents	Table of Contents	1
System Description	Overview Of The Keyport System	2
	Installing And Calibrating The Keyport	4
	Programmer Overlay	6
	Designing A Keyport Application	7
	SAMPLE Overlay Under Development	8
	Designing An Overlay	9
	DEFINE BOARD Screen	11
	Defining A Board	12
	Functional Flow Chart	16
	Designing Your Program	18
	Linking The Keyport Monitor	20
Reference	Advanced KEYPORT Programming	22
	KDB Structure Of The Basic Interface Special Functions	23
	Memory Map	25
	Memory Page 3 Usage	26
	KEYPORT Monitor Source Code	29
	SAMPLE Program Lo-Res Graphics	45
	SAMPLE Overlay	46



Introduction	<p>The KEYPORT 717 was designed with both the user and programmer in mind.</p> <p>For the user, the KEYPORT Overlay shows all the available program commands and options as pictures, labels and colors printed directly on and around the keys. Each command has its own key. With the KEYPORT, the user does <i>not</i> have to learn a new language to run an application. Program response time is shorter with fewer chances to make mistakes through syntax errors, wrong commands, etc. Typing of commands is eliminated, and user friendliness is greatly increased.</p> <p>For the application programmer, using the KEYPORT to design a specific overlay for user input to a program can eliminate the need for screen menus and interpretation of command syntax. This saves programmer time and money.</p> <p>The pressure sensitive membrane linked to KEYPORT software transmits specific information directly to the application program. This information consists of a function number necessary for the program to jump to the appropriate program subroutine and the set of input parameters required by the subroutine.</p>
Required Hardware	<p>The KEYPORT requires a 48K Apple II, II+ or IIe and a single disk drive. The KEYPORT does <i>not</i> require an interface card or power supply.</p>
Using Application Overlays	<p>This manual is designed to help you develop your own application overlays. You do <i>not</i> need to read this manual to use the KEYPORT with existing application overlays. The information needed to use an existing overlay package is contained in the application's Data Sheet.</p>
KEYPORT System	<p>The KEYPORT System consists of both hardware and software.</p> <p>The hardware includes:</p> <ul style="list-style-type: none"> <li>• Flat membrane keyboard with 717 keys</li> <li>• Programmer's overlay</li> <li>• Cable with connectors</li> </ul> <p>The software consists of:</p> <ul style="list-style-type: none"> <li>• KEYPORT Monitor (KP), a machine language program containing the utilities necessary to interface your application program and the KEYPORT system.</li> <li>• Key Data Table (KDT) with the Key Data Blocks (KDB's) containing the function number and input parameters assigned to each key. The KDT is defined using the DEFINE BOARD program supplied with the KEYPORT Monitor Diskette.</li> <li>• Your application program complete with its own overlay. Only the keys used in the program are printed on the overlay. Think of each overlay as a special input peripheral for an application.</li> </ul>

Illustration



Sample  
Application

We have included a **SAMPLE** lo resolution graphics application program and overlay in this manual. This application lets you draw shapes in low-resolution colors and save them on the diskette.

Related Pages

SAMPLE Program Listing, pg. 45  
SAMPLE Overlay, pg. 46

## INSTALLING AND CALIBRATING THE KEYPORT

### Apple IIe Restriction

Although the Apple IIe has 2 Game I/O sockets you can only use one at a time. Your KEYPORT will *not* work if the Apple IIe has something plugged into both the internal and external game connectors.

### WARNING

Before you connect the cable or any electrical equipment to your computer, always be sure the power is OFF. Otherwise, you could destroy a RAM chip or one of the other ICs.

### Calibration

When you calibrate the KEYPORT, the computer stores the X, Y values of all the keys of one column and one row. These values are used to identify any pressed key on the KEYPORT. You must calibrate the KEYPORT the first time you use a KEYPORT-computer combination.

After you have calibrated, everytime you boot the KEYPORT Monitor Diskette, the calibration data is automatically loaded with the KP Monitor.

You must have calibration data on every diskette you use with the KEYPORT.

### Restarting Calibration

If you press the wrong keys while calibrating the KEYPORT, you can restart the calibration by pressing **R**.

### Procedure

STEP	PROMPT	PROCEDURE
1.	—	<p>If your KEYPORT cable has a 9 pin connector, you can plug it into the rear panel of the Apple IIe.</p> <p>Otherwise, position the 16 pin dip connector over the Game I/O connector at the right rear of the Apple. Be sure the notched corner of the connector or the white dot is toward the <i>front</i> of the Apple.</p> <p>Gently push the pins into the socket until they are firmly seated.</p>
2.		Boot your KEYPORT Monitor Diskette.

Continued On Next Page



# INSTALLING AND CALIBRATING THE KEYPORT (continued)

Procedure

STEP	PROMPT	PROCEDURE
3.	KP MONITOR MENU	Enter <b>1</b> to calibrate a new KEYPORT-computer combination.
4.	<b>PRESS KEY OF Y COORD:</b>	Find the 12th <i>column</i> from the left on the KEYPORT. Beginning at the bottom row, press each key in the column. The computer beeps as you press each key. When you press the top (18th) key, the computer beeps twice to let you know it is ready to calibrate the X coordinates.
5.	<b>PRESS KEY OF X COORD:</b>	Find the 9th <i>row</i> from the bottom on the KEYPORT. Beginning at the left, press each key in the row. The computer beeps as you press each key. When you press the last (44th) key, the system saves the calibration data onto the diskette.
6.	KP MONITOR MENU	If you want to copy the calibration data onto another diskette, enter <b>3</b> and go to Step 7.  Otherwise, press <b>RETURN</b> . The system loads the KP Monitor and displays the AppleSoft BASIC prompt.
7.	<b>INSERT A DISKETTE AND PRESS RETURN FOR A COPY OR ANY KEY TO QUIT</b>	Put the diskette into your drive and press <b>RETURN</b> . You can copy the calibration data onto as many diskettes as you want.  When you have made all of the calibration copies you want, press any key <i>except</i> <b>RETURN</b> . The system displays the AppleSoft BASIC prompt.

KEYPORT  
Does Not  
Respond

If you take the KEYPORT with you to a different city or if the temperature changes, the KEYPORT-computer calibration might change. If your KEYPORT does *not* appear to respond, recalibrate and try again.

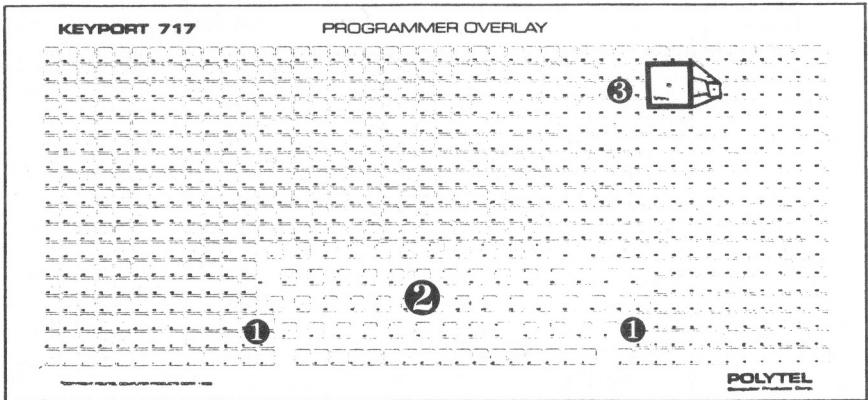
Backup  
Copies

Be sure you make a backup copy of your KEYPORT Monitor Diskette.

## Description

The Programmer Overlay helps you to design an application for the KEYPORT. The overlay tells you the location of each key so you can design your own customized keyboard. Although a section of keys is set up with typewriter spacing, you do *not* have to define them as typewriter keys. *Any* key can be programmed the way you want.

## Sample



## PROGRAMMER OVERLAY

## Parts

1. Typewriter Shift (S2)
2. Typewriter Keyboard
3. Key Location Number

## Introduction

Designing an application program using the KEYPORT system is a simple process. You can design and write programs faster without having to worry about designing menus or setting up complex syntax analysis.

Simply follow the design method outlined in this manual.

## Suggested Design Method

STEP	PROCEDURE	SEE PAGE
1.	Design the overlay. Assign a function number and input parameters to each key or group of keys.	9
2.	Define your board and create a Key Data Table (KDT) containing your key functions and parameters.	12
3.	Develop a Functional Flow Chart for your application.	16
4.	Design and write your application program.	18
5.	Link the KP Monitor to your application program.	20

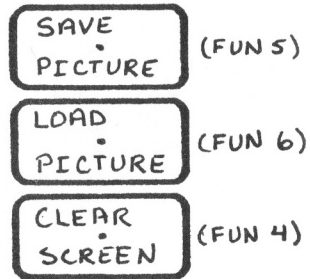
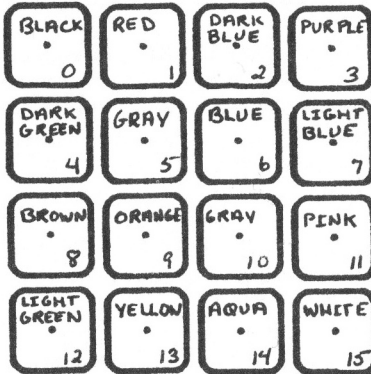


Description

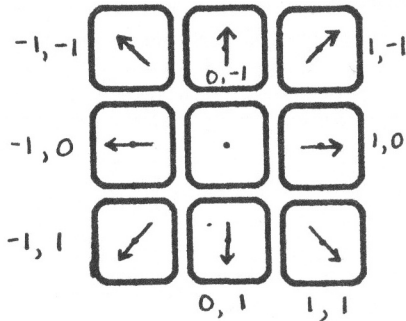
Here is a copy of the SAMPLe Overlay during development. Notice that we have written down the function numbers and string of input parameters for each key.

Sample

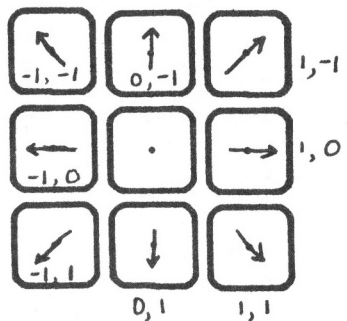
CURSOR COLOR (FUN 3)



MOVE (FUN 1)



MOVE & PLOT (FUN 2)



## Introduction

Before you define a KEYPORT application decide what functions you want, what keys you need and how you want each key displayed on the overlay.

Materials  
Required

- Programmer Overlay
- Tracing Paper (optional)

Sample  
Application

The example in this procedure is a color graphics application. This application lets you draw shapes in low-resolution color graphics and save them on the diskette. This application is included on your KP Monitor Diskette under the name **SAMPLE**.

## Procedure

STEP	PROCEDURE
1.	<p>List all the commands you want to show on the overlay.</p> <p>For example:</p> <ul style="list-style-type: none"> <li>• Move cursor <i>without</i> plotting in 8 directions</li> <li>• Move cursor <i>with</i> plotting in 8 directions</li> <li>• Change the color of the cursor to a different lo-res color</li> <li>• Clear the screen</li> <li>• Save the picture</li> <li>• Load a picture from the diskette</li> </ul>
2.	<p>Decide what commands ought to be grouped under a single function.</p> <p>For example:</p> <ol style="list-style-type: none"> <li>1. All 8 moves <i>without</i> plot</li> <li>2. All 8 moves <i>with</i> plot</li> <li>3. Cursor color changes</li> <li>4. Clear screen</li> <li>5. Save picture</li> <li>6. Load picture</li> </ol>

Continued On Next Page

## DESIGNING AN OVERLAY (continued)

Procedure

STEP	PROCEDURE
3.	Decide where you want each command on the overlay. You can assign the same command to 2 or more locations.
4.	Sketch the location of each key on the Programmer Overlay or the tracing paper. Use a soft lead pencil, a grease pencil or an alcohol-based felt-tipped pen.
5.	Print the name or explanation of each command on the Programmer Overlay.
6.	<p>Assign a number to each function and print it on the corresponding keys.</p> <p>For example:</p> <ul style="list-style-type: none"> <li>• Move — Function 1</li> <li>• Colors — Function 3</li> <li>• Save — Function 5</li> <li>• Plot — Function 2</li> <li>• Clear — Function 4</li> <li>• Load — Function 6</li> </ul>
7.	<p>For each key, print the input parameter assigned to the key. The parameters are the information used by the subroutine to perform the specific function of the key.</p> <p>For example, for the group of keys having Function 3, Change Colors, the input parameters should be the Apple II low-resolution color codes:</p> <p style="margin-left: 40px;">0 for black 1 for magenta 2 for dark blue, etc.</p> <p>For the group of keys having Function 1, Move Cursor, the input parameters should be the change in coordinates X and Y:</p> <p style="margin-left: 40px;">1,0 to move right -1,0 to move left 0,1 to move down, etc.</p>

When You  
Finish

When you have the overlay designed, use the procedure **DEFINING A BOARD** to set up your Key Data Blocks.

Related Pages

Defining A Board, pg. 12



## Description

The **Define Board Screen** is designed to assign function numbers and input parameters to individual KEYPORT keys.

## Diagram

**BOARD\_DEFINITION\_PROGRAM**

**1** BOARDNAME :

KEY : **2**  
 FUNCTION : **3**  
 ASCII CODE : **4**  
 STRING : **5**

0-255 **6**

---

E=EDIT - D=DELETE  
 P=DUPLICATE  
 S=SAVE - Q=QUIT

**7**

<ESC> TO ENTER  
 A NUMBER FROM  
 0-255

---

ENTER COMMAND OR PRESS KP KEY

**8**

---

## SCREEN DURING DEFINE BOARD UTILITY PROGRAM

### Parts

- |                                       |                               |
|---------------------------------------|-------------------------------|
| 1. Board Name                         | 5. String Of Input Parameters |
| 2. Key Number And Shift Status        | 6. Value Entry Field          |
| 3. Function Number                    | 7. Commands                   |
| 4. ASCII Code Of The Input Parameters | 8. Command Entry              |

Introduction	When you define a KEYPORT board, you actually assign each key a <b>Key Data Block</b> . The set of KDBs for each board is called the Key Data Table (KDT). The KDT is a major part of the communication between the KEYPORT and your program.
Key Data Block.	The <b>Key Data Block</b> (KDB) contains the function number and string of input parameters you assigned to each KEYPORT key. The KDB can act like a MACRO instruction you can <b>CALL</b> by pressing the key.
Shifting Keys	<p>Each key can be assigned 2 different KDBs by using the shift key. For example, you can assign 2 functions to key number <b>5</b> by pressing:</p> <ul style="list-style-type: none"><li>• Key number <b>5</b> without shift OR</li><li>• Typewriter shift (S2) and key number <b>5</b></li></ul>
Shift Status	<p>When you press a KEYPORT key, the key number and shift status are displayed:</p> <ul style="list-style-type: none"><li>• <b>NS</b> is no shift</li><li>• <b>S2</b> is typewriter keyboard shift</li></ul>
Entering Edit Keys	<p>While editing the string of input parameters for a given key, you might want to include characters like <b>ESC</b> or <b>→</b>. These keys are used in the editing process and can not be entered directly into a string. Instead we have defined the character <b>]</b> as an escape character to let you enter these editing keys.</p> <p>To enter a <b>]</b>, <b>ESC</b>, <b>→</b>, <b>←</b> or <b>RETURN</b>, enter <b>]</b>, (<b>SHIFT M</b> on the Apple II and <b>II+</b>) followed by the character. For example, to enter a <b>RETURN</b>, press <b>]</b>, then <b>RETURN</b>. To enter a <b>]</b>, press <b>]</b> twice.</p>
System Lockup	<p>When you are loading or saving a large board, the system may take a few minutes. It displays the key number it is processing for you.</p> <p>If the system seems to hang up while you are defining a board, wait a few seconds for the system to do its housekeeping.</p>
Before You Begin	<ul style="list-style-type: none"><li>• Use the procedure <b>DESIGNING AN OVERLAY</b> to decide on the function number and string of input parameters for each key you intend to use</li><li>• Boot your KP Monitor Diskette</li><li>• If you are using an Apple IIe, be sure the <b>CAPS LOCK</b> is down, so all your entries are in upper case</li></ul>

---

Continued On Next Page

Related Pages	Designing An Overlay, pg. 9 Linking The KEYPORT Monitor, pg. 20
---------------	--

## DEFINING A BOARD (continued)

Procedure

STEP	FIELD	PROCEDURE
2.	<b>BOARDNAME:</b>	<p>If you are changing an existing board, enter the name of the board you want to change. The system adds <b>BTAB</b> to the name you entered and loads the KDT of the board.</p> <p>If you are entering a new board, enter the name you want assigned to the board. The name can be any legal file name.</p>
3.	<b>ENTER COMMAND OR PRESS KP KEY</b>	<p>If you want to add, change, duplicate or delete a key, press the key. The system displays the key number and its shift status. Go to Step 4.</p> <p>If you want to save this board, enter <b>S</b> and go to Step 8.</p> <p>If you want to return to AppleSoft without saving this board, enter <b>Q</b> and go to Step 8.</p>
4.	<b>ENTER COMMAND OR PRESS KP KEY</b>	<p>If you want to add or change this key, enter <b>E</b> and go to Step 5. Do <i>not</i> press <b>RETURN</b>.</p> <p>If you want to delete this key, enter <b>D</b> and go to Step 3. The program deletes the function and string of input parameters from this key.</p> <p>If you want to duplicate this key function number and string of input parameters at another key location, enter <b>P</b> and press the new key. Go to Step 3.</p>
5.	<b>FUNCTION</b>	<p>Enter the function number for this key. Press <b>RETURN</b> and go to Step 6.</p>

Continued On Next Page

Related Pages

Shift Status, pg. 12

Procedure

STEP	FIELD	PROCEDURE
6.	<b>STRING</b>	<p>Enter the string of input parameters you want this key to send to your program. If you want to enter a binary value or an ASCII character <i>not</i> shown on the standard keyboard, go to Step 7.</p> <p>The ASCII value for each character is displayed in the ASCII CODE field as the cursor moves over the input parameter characters.</p> <p>You can use the arrow keys to move the cursor to the right and left through the set of input parameters. When you are satisfied with the string, move the cursor just to the right of the last character in your input parameters. Press <b>RETURN</b> and go to Step 3.</p>
7.	<b>0-255</b>	<p>Press <b>ESC</b> , enter the numeric value you want entered in the input parameters. Press <b>RETURN</b> . The program displays the ASCII character, if any, in the input parameters. Go to Step 6.</p>
8.	<b>DO YOU WANT TO QUIT?</b>	<p>If you do <i>not</i> want to quit, enter <b>N</b> and go to Step 3.</p> <p>Otherwise, enter <b>Y</b>. The program erases all entries and displays the AppleSoft prompt.</p>

Continued On Next Page

Related Pages

Character Display, pg. 15  
Entering Edit Keys, pg. 12

## DEFINING A BOARD (continued)

### Procedure

STEP	FIELD	PROCEDURE
9.	<b>DO YOU WANT TO SAVE?</b>	If you do not want to save this board, enter <b>N</b> , and go to Step 3. Otherwise, enter <b>Y</b> and the program processes the KDT. The screen displays the number of keys processed. The longer your KDB's the longer this will take. Go to Step 10.
10.	<b>FILE:</b>	Enter the name you want assigned to this Key Data Table. The system saves the KDT on the diskette and displays the total length of your KDT in decimal bytes. If this number exceeds 8192 bytes, your program will <i>not</i> work. Reload your KDT and shorten some of your input parameters.

### Character Display

The system displays:	As:
Normal alphanumeric character	The normal character
← OR →	A flashing < OR >
ESCAPE	A flashing <b>E</b>
RETURN	A flashing <b>R</b>
Control Character	An inverse character
Characters with an ASCII CODE above 127	An inverse question mark (?)

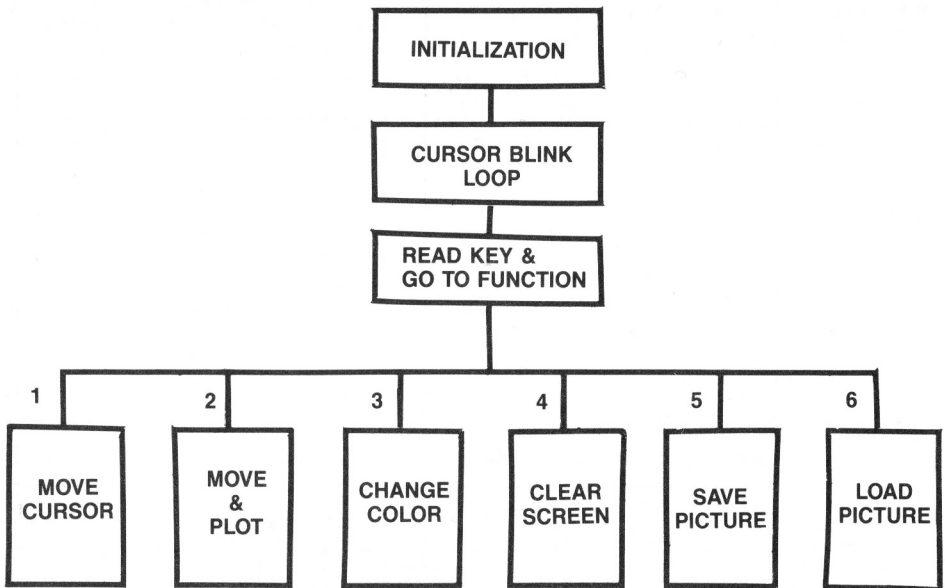
### Number of Keys

If you count the total number of keys on your KEYPORT you will find either 717 or 720 key locations; however, when you load a KDT you will notice 792 keys are loaded. This is because the Key Data Table is an 18 X 44 matrix (18 X 44 = 792). Since the typewriter keys use more space than the other keys, the KEYPORT has only 717 (or 720) key locations.

## Introduction

This Functional Flow Chart shows the design of the **SAMPLE** Graphics Application. You can use it as an example to design your own application.

## Flow Chart



**FLOW CHART FOR LO-RES GRAPHICS SAMPLE PROGRAM**

Continued On Next Page

# FUNCTIONAL FLOW CHART (continued)

Functional  
Flow

STEP	PROCESS	EXAMPLE
1.	Initialization routine	<ul style="list-style-type: none"> <li>• Set screen to low-resolution graphics</li> <li>• Set cursor to screen center</li> <li>• Branch to cursor blink routine</li> </ul>
2.	Do any loop routines and check for input	<ul style="list-style-type: none"> <li>• Display cursor in current color</li> <li>• Erase cursor</li> <li>• Check if KEYPORT key pressed</li> </ul>
3.	If KEYPORT key is pressed, get function number and go to subroutine	<ul style="list-style-type: none"> <li>• PEEK to get function number</li> <li>• Branch to subroutine</li> </ul>
4.	Execute subroutine. Go to the loop routine (Step 2) and check for additional input	<ul style="list-style-type: none"> <li>• PEEK (513), (514), (515) etc. to get input parameters, for the subroutine</li> <li>• Perform subroutine function, i.e. move cursor, change color, etc.</li> <li>• Branch to loop routine</li> </ul>

Related Pages

PEEK Input Parameters, pg. 19  
PEEK Function Number, pg. 19

Program  
Structure

We recommend your applications contain:

- An initialization routine
  - A read routine
  - A set of function subroutines
  - Any optional loop routines necessary
- 

## How It Works

When your program requests data (INPUT, GET) the KEYPORT Monitor looks at both the standard computer keyboard and KEYPORT connected to the Game Port. When the user presses any key the Monitor sets a flag. By PEEKing at the flag, you can determine where the input came from. When a KEYPORT key is pressed, the Monitor puts the key function number and input parameters into memory so your program can read and use them.

If you did *not* load a KDT, the KP Monitor puts only the KEYPORT key number into memory at locations 774 and 775 decimal.

---

Function  
Subroutine

A *function subroutine* is the part of your application that performs the action you indicated on a KEYPORT key. A function subroutine gets the input parameters from the KDB and uses these parameters to accomplish the purpose of the subroutine.

---

## Loop Routine

You might also want to include a *loop routine* in your program. The loop can do any necessary repetitive functions like blinking the cursor.

---

KDB  
Limitations

You can have up to 255 different function numbers. Each key can have up to 250 ASCII characters in its KDB string of input parameters.

---

Length Of  
KDB

The length of the KDB is the length of input parameters *plus* 1 byte for the function number.

For example, if the string of input parameters is ABC, the KDB length is 4. If the string is PRINT "HELLO", the KDB length is 14. The length of the KDB is stored in an extra byte just before the function number.

---

Continued On Next Page



Memory  
Locations

If you want to:	Then:
Check to see if a key has been pressed	CALL 786, then PEEK (768) 0 = No key pressed 1 = KEYPORT key 2 = Keyboard key
Get the function number (FUN) of a KEYPORT key	PEEK(779)
Get KDB input parameters	PEEK(513) PEEK(514) PEEK(515), etc. for each character in the string
Get the length (LEN) of the KDB of the key pressed. In some cases, your program does <i>not</i> need this length.	PEEK(780)
Get the number of the KEYPORT key	PEEK(774) + 256 *PEEK(775)
Get the shift switch status	PEEK(776) 0 = No Shift 1 = Shift
Get the ASCII code of a keyboard key that was pressed	PEEK(49152), \$C000 HEX  See your Apple Reference Manual for a list of ASCII codes.

Related Pages

Memory Page 3 Usage, pg. 26

Description	<p>The KEYPORT Monitor is the machine language program linking the KEYPORT and your program.</p> <p>When you boot the KEYPORT Monitor Diskette, it loads the KP Monitor and your Key Data Table (KDT).</p>
Loading The Key Data Table	<p>When you boot KP Monitor, it automatically loads the KDT listed in line 10 of the HELLO program. Initially line 10 is <b>DATA SAMPLEBTAB</b>. If you do <i>not</i> want a KDT loaded, change line 10 to <b>DATA NO</b>.</p> <p>If you want a <i>different</i> KDT loaded, change line 10 to the name of your KDT, for example 10 <b>DATA MINEBTAB</b>. Remember, the <b>DEFINE BOARD</b> program adds <b>BTAB</b> to every KDT file name.</p>
Monitor Location	<p>The KP Monitor can be loaded immediately below DOS. If you have an Apple IIe or an Apple II with a language card, the KP Monitor can be loaded in the top 16K of memory.</p> <p>If you want the KP Monitor loaded <i>below</i> DOS, change the KP Monitor <b>HELLO</b> program line 20 to <b>DATA MEM</b>.</p> <p>If you want the KP Monitor loaded in the top 16K of memory, change line 20 to <b>DATA AUTO</b>. However, the system checks to be sure you have a 64K system before trying to load the Monitor into the top 16K. If you have a 48K machine, it <i>automatically</i> loads the Monitor below DOS. See <b>MEMORY MAP</b> for more detail.</p>
KP Monitor Menu	<p>If you do <i>not</i> want to display the KP Monitor Menu, change line 30 to <b>DATA NO</b> and the system will load the Monitor and your KDT automatically.</p> <p>If you want to display the KP Monitor Menu, leave line 30 <b>DATA YES</b> so the system displays the program choices for you.</p>
Calibration	<p>You must calibrate the KEYPORT the first time you use a particular KEYPORT-computer combination. For more information about calibration, see the procedure <b>INSTALLING AND CALIBRATING THE KEYPORT</b>.</p>

Continued On Next Page

## LINKING THE KEYPORT MONITOR (continued)

### Procedure

STEP	PROMPT	PROCEDURE
1.	—	Boot the KP Monitor Diskette.
2.	KP MONITOR MENU	<p>If you want to load the KP Monitor, press <b>RETURN</b> .</p> <p>If you want to calibrate a new KEYPORT or copy calibration data onto another diskette, see <b>INSTALLING AND CALIBRATING THE KEYPORT</b>.</p> <p>If you want to get out of this program, press <b>4</b> .</p> <p>If you want to define a KEYPORT board, press <b>2</b> . See <b>DEFINING A BOARD</b>.</p>
3.	AppleSoft BASIC Prompt ]	The system has loaded the KP monitor and your KDT.

### When You Finish

Whenever you issue an **INPUT** or **GET** command, the KP Monitor reads both the Keyboard and the KEYPORT. If you loaded a KDT, whenever you press a KEYPORT defined key, the KDB information is returned to your program.

### Multiple KDT's On The Same Diskette

You can save and use more than one KDT. Simply **LOAD** the **HELLO** program, change the **DATA** statement in line 10, as described above, and save the **HELLO** program under a different name. See **LOADING THE KEY DATA TABLE** and **MONITOR LOCATION**.

To **LOAD** the alternate KDT and the KP Monitor, **RUN** the new program under its own name.

### Related Pages

Installing And Calibrating The KEYPORT, pg. 4  
Defining A Board, pg. 12  
Loading The Key Data Table, pg. 20  
Monitor Location, pg. 20

## Introduction

If you have a BASIC Overlay you can use the USER DEFINED KEYS to output KDB information to a BASIC program running under the control of the BASIC Interface rather than the KEYPORT Monitor. By using this procedure, you can define a simple board *without* using the DEFINE BOARD program.

---

## SAMPLE Program

To use the SAMPLE program with the Basic Overlay:

- Tape your SAMPLE Overlay over the USER DEFINED KEYS in the upper left corner of your BASIC Overlay so the bottom left key of the MOVE CURSOR section is over the bottom left white User Defined Key.
  - Boot your BASIC Interface Diskette
  - LOAD BSAMPLBTAB using the LOAD BOARD key
  - RUN BSAMPLE
- 

## Procedure

If you want to use the BASIC Overlay to output KDB information, simply:

- Press **DEFINE CHARACTER STRING** or **DEFINE KEY SEQUENCE**
- Press the USER DEFINED KEY you want to use
- Enter the function number for the key and **RETURN**
- Enter the input parameter you want this key to send to your program followed by a **RETURN**
- Press **END OF DEFINITION**

For example, we defined the keys in BSAMPLBTAB with the following keys:

**DEFINE  
CHARACTER  
STRING**



**1 RETURN 0,1 RETURN END OF DEFINITION**

---

## When Running BSAMPLE

When the program issues an INPUT F statement, if the user presses this key, F contains 1. The program will GOTO the subroutine for Function 1. Then the subroutine issues an INPUT DX,DY statement, the system will put 0 into DX and 1 into DY.

---

## Redefining The Board

You can also use the DEFINE BOARD program to redefine ALL of the keys on the BASIC Programming Overlay. When the system prompts for the board name, enter BASIC. The program automatically adds BTAB to this name. See KDB STRUCTURE OF THE BASIC INTERFACE SPECIAL FUNCTIONS for the KDB formats.

---

## Related Pages

KDB Structure of the BASIC Interface Special Functions, pg. 23

# KDB STRUCTURE OF THE BASIC INTERFACE SPECIAL FUNCTIONS

FUNCTION	DESCRIPTION	KDB FORMAT
0	Generate Key Sequence	<div> <div>LENGTH</div> <div>0</div> <div>KEY 1</div> <div>KEY 2</div> <div>KEY N</div> <div>FF FF</div> </div> <div>delimiter</div>
1	Generate String Of Characters	<div> <div>LENGTH</div> <div>1</div> <div>STRING OF CHARACTERS</div> </div>
2	USER DEFINED KEY	<div> <div>length</div> <div>←</div> <div>→</div> <div> <div>LENGTH</div> <div>2</div> <div>1</div> <div>len*</div> <div>ANY CHARACTER STRING</div> </div> <div>←</div> <div>actual length len*</div> <div>→</div> <div>1 = defined as a character string</div> <div>OR</div> <div> <div>LENGTH</div> <div>2</div> <div>0</div> <div>KEY1</div> <div>KEY2</div> <div>KEY3</div> <div>...</div> <div>FF FF</div> </div> <div>0 = defined as a key sequence</div> </div>
3	NOT USED	
4	Define USER DEFINED KEY As A Character String	<div> <div>LENGTH</div> <div>FUNCTION</div> <div>1</div> <div>4</div> </div>
5	Define USER DEFINE KEY As A Key Sequence	<div> <div>LENGTH</div> <div>FUNCTION</div> <div>1</div> <div>5</div> </div>
6	End Of Definition Of USER DEFINED KEY	<div> <div>LENGTH</div> <div>FUNCTION</div> <div>1</div> <div>6</div> </div>

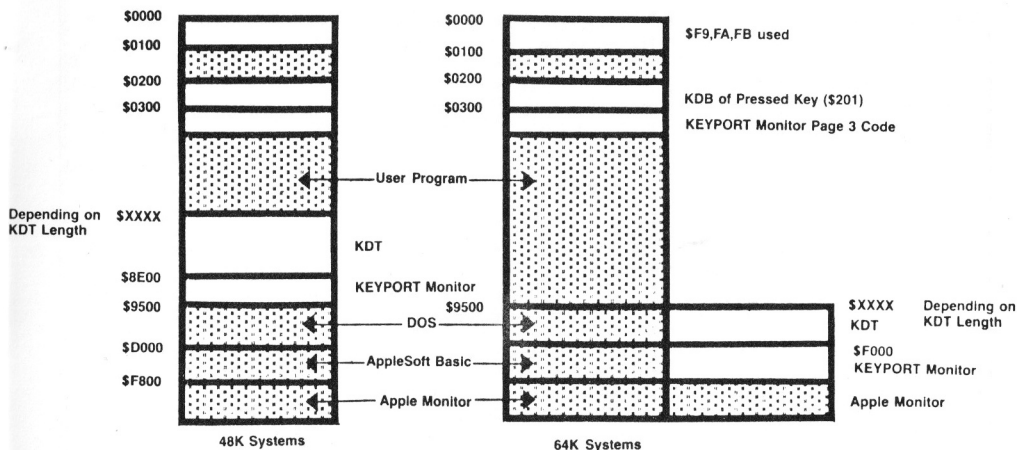
**KDB STRUCTURE OF THE BASIC INTERFACE SPECIAL FUNCTIONS**  
(continued)

7	Save Board	LENGTH      FUNCTION 1              7
8	Load Board	LENGTH      FUNCTION 1              8
16	<b>RETURN &amp; NEW LINE NUMBER</b>	LENGTH      FUNCTION 1              16
17	<b>DEFINE LINE NUMBER STEP</b>	LENGTH      FUNCTION 1              17
18	<b>DEFINE LINE NUMBER START</b>	LENGTH      FUNCTION 1              18
19	<b>STOP LISTING</b>	LENGTH      FUNCTION 1              19
20	<b>RESUME LISTING</b>	LENGTH      FUNCTION 1              20
21	<b>LIST ALL</b>	6              21      L   I   S   T   CR
22	<b>ABORT LISTING</b>	LENGTH      FUNCTION 1              22

Description

The KEYPORT Monitor and Key Data Table, if one exists, are loaded by the HELLO program. In a 48K system, they are loaded immediately below DOS. In a 64K system, they are loaded in the top 16K of memory unless line 20 is DATA MEM.

Diagram



KP MONITOR MEMORY MAP

\$XXXX

\$XXXX is calculated by HELLO based on the length of the KDT. The length of the KDT is recorded in the first 2 bytes of the KDT by the DEFINE BOARD program. In a 48K system, HIMEM is set to \$XXXX.

Maximum Length Of KDT

The maximum length of a KDT is 2000 hexadecimal or 8192 decimal. The program DEFINE BOARD gives you the length of the KDT after saving the table. If the KDT is longer than 8192, you *must* reduce the size by either eliminating some of the keys or short-tening the length of some of the input parameters.

Page 2 Usage

\$201 forward (513 decimal) contains the Key Data Block input parameters of the last KEYPORT key pressed.

Related Pages

Loading The Key Data Table, pg. 20  
Monitor Location, pg. 20

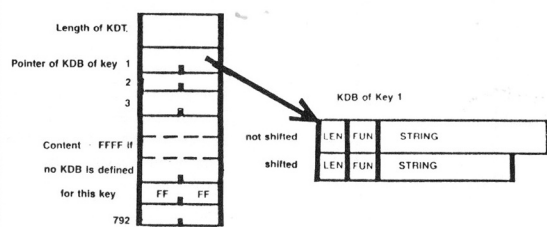
Updated  
Location

Memory locations \$300 to \$30C (768 to 780 decimal) are automatically updated every time a GET, INPUT or CALL to RDKEY (CALL 64780) is issued.

HEX	DECIMAL	SYMBOLIC NAME	DESCRIPTION
\$300	(768)	RTFL	<p>After a call to \$312 (786) to test if a key has been pressed, this byte contains:</p> <ul style="list-style-type: none"> <li>• 0 if <i>no</i> key was pressed.</li> <li>• 1 if a KEYPORT key was pressed.</li> <li>• 2 if a keyboard key was pressed. You must reset the Keyboard Strobe by issuing the command POKE 49168,0.</li> </ul> <p>After a GET or INPUT or a CALL to the Apple Monitor routine RDKEY (\$FDOC), 64780 decimal, this byte contains:</p> <ul style="list-style-type: none"> <li>• 0 if keyboard or KEYPORT key was pressed. See \$306/307 for the key number.</li> <li>• 3 if the key was pressed and <i>no</i> KDB has been assigned to this key.</li> </ul>
\$301	(769)	RY	The measured resistance of the Y coordinate.
\$302 /303	(770/771)	RX	The measured resistance of the X coordinate, least significant byte/most significant byte.
\$304	(772)	ROW	The row number of the KEYPORT key pressed from the bottom up, number 1 to 18.
\$305	(773)	COLUMN	The column number of the KEYPORT key pressed from left to right, number 1 to 44.
\$306 /307	(774/775)	KEY	<p>Key number of KEYPORT key pressed, least significant byte/most significant byte, from 1 to 792.</p> <p><math>KEY = (ROW - 1) * 44 + COLUMN</math></p> <p>If a keyboard key is pressed, these bytes are 0.</p>

Continued On Next Page



HEX	DECIMAL	SYMBOLIC NAME	DESCRIPTION
\$308	(776)	SHIFT	The shift status of a KEYPORT key. If this byte contains: <ul style="list-style-type: none"> <li>• 0 key is <i>not</i> shifted</li> <li>• 1 key <i>is</i> shifted</li> </ul>
\$309 /30A	(777/778)	ADKDB	Pointer to the byte preceding the KDB of the key pressed, least significant byte/most significant byte. The byte pointed to contains LEN, the length of this KDB including FUN, the function number byte. The next byte contains FUN, the function number (1 byte) followed by the string of parameters of length LEN-1 bytes.
\$30B	(779)	FUN	The function number of the pressed key.
\$30C	(780)	LEN	The length of the KDB of the key pressed.
\$30D /30E	(781/782)	TABD	<p>Pointer to the third byte of the Key Data Table (KDT), least significant byte/most significant byte. The first 2 bytes of the KDT are the length of the KDT. TABD points to the <i>third</i> byte of the KDT which is the start of a look-up table of 792 addresses, each 2 bytes long. These addresses are relative pointers to the KDB's of the corresponding keys.</p> 

Continued On Next Page

HEX	DECIMAL	SYMBOLIC NAME	DESCRIPTION
\$30F /310	(783/784)	ADCALIB	Address of the Calibration Table, least significant byte/most significant byte.
\$311	(785)	RDFL	<p>When a GET or INPUT is issued, if this byte contains:</p> <ul style="list-style-type: none"> <li>• 1 the KEYPORT Monitor calculates the key number of the KEYPORT key pressed</li> <li>• 0 the KEYPORT Monitor calculates the key number of the KEYPORT key pressed and gets the KDB</li> </ul> <p>This byte can be set by the user.</p>
\$312 /313 /314	(786/ 787/ 788)	CALL	A CALL to 786 causes a jump to the KP Monitor routine that checks whether a keyboard or KEYFORT key was pressed. The code or "flag" returned by this routine is stored in \$300 (See \$300).
\$315	(789)	CHAR	<p>Initially this byte contains \$8D (carriage return). When a GET or INPUT is issued and a KEYPORT key is pressed, the KEYPORT Monitor puts the contents of this byte in the accumulator before returning to the CALLing program.</p> <p>You can change the contents of \$315 to a value between B0 and B9 (176 to 185 decimal) by issuing POKE 789,176, before issuing a GET X statement to avoid a BASIC program syntax error when you use a GET X statement with the KEYPORT.</p> <p>If you use GET X\$, instead of GET X, there will be <i>no</i> syntax error, but the BASIC compiler will use more memory, because each string variable, like X\$, used with GET is assigned its own memory space, while a numeric variable like X is <i>not</i>.</p>

0800	1	*****
0800	2	**
0800	3	**
0800	4	**
0800	5	**
0800	6	**
0800	7	**
0800	8	**
0800	9	**
0800	10	**
0800	11	**
0800	12	**
0800	13	**
0800	14	**
0800	15	**
0800	16	**
0800	17	**
0800	18	**
0800	19	**
0800	20	**
0800	21	*****

KEYPORT 717 UTILITY DISKETTE

COPYRIGHT 1983 POLYTEL COMPUTER PRODUCTS CORP.

AUTHORS SHERIF DANISH AND KRIS KIMBROUGH

THESE UTILITIES ARE CALLED EVERY TIME A READ REQUEST IS ISSUED TO THE APPLE MONITOR THROUGH THE SUBROUTINE IN \$38. THIS CODE LOOPS ON THE KEYBOARD AND THE KEYPORT UNTIL A KEY IS PRESSED. IF A KEYPORT KEY IS PRESSED THE RESISTIVE STRIPS ARE READ AND THE KEY NUMBER IS DERIVED. IF THE KEY HAS A KDB THE STRING OF THE KEY PRESSED IS STORED STARTING AT \$201 THROUGH \$2FF AS NEEDED. THE FUNCTION NUMBER IS PUT IN \$30B. THE KDB LENGTH CAN BE FOUND IN \$30C. THE ACCUMULATOR RECIEVES THE CHARACTER FOUND IN \$315. IF A KEYBOARD KEY IS PRESSES THEN THAT CHARACTER IS RETURNED IN THE ACCUMULATOR.

0800	22	***** USER'S DATA	
0800	23	*	
0300	24	DATA	ORG \$300
0300 00	25	RTFL	HEX 00
			:RETURN FLAG
0301 00	26	RY	HEX 00
			:RES. OF Y
0302 00 00	27	RX	HEX 0000
			:RES. OF X
0304 00	28	RQW	HEX 00
0305 00	29	COLUMN	HEX 00
0306 00 00	30	KEY	HEX 0000
0308 00	31	SHIFT	HEX 00
			:SHIFT FLAG
0309 00 00	32	ADKDB	HEX 0000
			:ADDR OF KEY DATA BLOCK (KDB)
030B 00	33	FUN	HEX 00
			:FUNCTION NUMBER
030C 00	34	LEN	HEX 00
			:KDB LENGTH
030D 00 00	35	TABD	HEX 0000
			:DATA TABLE LOC.
030F 0F 94	36	ADCALIB2	ADR CALTAB
0311 00	37	RDFL	HEX 00
0312 4C FB 91	38	JMP KPON	
0315 8D	39	CHAR	HEX 8D
			:KEYPORT RET.WITH THIS CHAR.
0316 01 00	40	N	HEX 0100
0318 22	41	WT	HEX 22
0319	42	*	
0319	43	*	
8E00	44	ORIGIN	EQU \$8E00
8E00	45		ORG ORIGIN
8E00 4C 29 91	46		JMP INIT
C062	47	S2SWITCH	EQU \$C062
C063	48	S3SWITCH	EQU \$C063
8E03 00 00	49	LASTREAD	HEX 0000
			:LAST RES.READ
8E05 0F 94	50	ADCALIB	ADR CALTAB
8E07	51	*	
8E07	52	***** TEST KEYBOARD & KEYPORT	
8E07	53	*	
8E07 8D DB 91	54	START	STA Z
8E0A 8E D9 91	55		STX Z+1
8E0D 8C DA 91	56		STY Z+2
8E10 AD 16 03	57		LDA N
8E13 8D 17 03	58		STA N+1
8E16 E6 4E	59	KEYIN	INC \$4E
			:INCR RANDOM NUMBER
8E18 D0 02	60		BNE KEYIN2
8E1A E6 4F	61		INC \$4F
8E1C 2C 00 C0	62	KEYIN2	BIT \$C000
			:KEYBOARD KEY ON? :
8E1F 30 12	63		BMI KEYIN4
			:YES
8E21 20 FB 91	64		JSR KPON
			:NO,IS KP KEY PRESSED ?
8E24 B0 2A	65		BCS REPEAT
			:YES,DO YOU WANT TO REPEAT
8E26 A9 00	66	KEYIN3	LDA #\$00
			:NO KEY PRESSED
8E28 8D D7 91	67		STA FLAG
			:CLEAR REPT FLAG
8E2B A9 40	68		LDA #\$40
8E2D 8D D6 91	69		STA CCOUNT
8E30 4C 16 8E	70		JMP KEYIN
8E33 A9 00	71	KEYIN4	LDA #\$00
			:SET RETURN FLAG & KEY TO 0
8E35 8D 00 03	72		STA RTFL
			:TO SHOW THAT A KEYBOARD
8E38 8D 06 03	73		STA KEY
			:KEY WAS PRESSED
8E3B 8D 07 03	74		STA KEY+1
8E3E AC DA 91	75		LDY Z+2
8E41 AE D9 91	76		LDX Z+1
8E44 AD D8 91	77		LDA Z
8E47 91 28	78		STA (\$28),Y
8E49 AD 00 C0	79		LDA \$C000

8E4C	2C 10 C0	80	BIT #C010	
8E4F	60	81	RTS	;RETURN WITH CHR IN ACCUM.
8E50		82	*	
8E50		83	***** REPEAT ?	
8E50		84	*	
8E50	AD D7 91	85	REPEAT LDA FLAG	;IS FLAG=0,NO REPEAT
8E53	F0 17	86	BEQ REPEAT4	;YES,INCR.FLAG & READ
8E55	AD D6 91	87	LDA CCOUNT	;HAVE WE WAITED 1 SEC ?
8E58	F0 18	88	BEQ STARTC	;YES,READ IMMEDIATELY
8E5A	A9 40	89	REPEAT1 LDA #40	;NO,THEN WAIT
8E5C	20 A8 FC	90	JSR #FCAB	
8E5F	20 F8 91	91	JSR KFOR	;IS KEY STILL PRESSED ?
8E62	90 C2	92	BCC KEYIN3	;NO RESET COUNTER
8E64	CE D6 91	93	DEC CCOUNT	
8E67	D0 F1	94	BNE REPEAT1	;THROUGH ?
8E69	4C 72 8E	95	JMP STARTC	;YES,READ
8E6C	EE D7 91	96	REPEAT4 INC FLAG	;INC FLAG
8E6F	4C 72 8E	97	JMP STARTC	;GO READ
8E72		98	*	
8E72		99	***** PROGRAM START	
8E72		100	*	
8E72	A9 00	101	STARTC LDA #00	;RESET RETURN FLAG
8E74	8D 00 03	102	STA RTFL	
8E77	20 D3 8E	103	JSR READKEY	;READ KEY & GET KEY #
8E7A	AD 00 03	104	LDA RTFL	
8E7D	C9 FF	105	CMP #FF	;VALID READING?
8E7F	D0 20	106	BNE STARTC1	
8E81	4C 1C 8E	107	JMP KEYIN2	;BAD READ SO LOOP
8E84	EE 04 03	108	STARTC2 INC ROW	
8E87	EE 05 03	109	INC COLUMN	
8E8A	EE 06 03	110	INC KEY	
8E8D	D0 03	111	BNE STARTC3	
8E8F	EE 07 03	112	INC KEY+1	
8E92	AD D8 91	113	STARTC3 LDA Z	
8E95	AE D9 91	114	LDX Z+1	
8E98	AC DA 91	115	LDY Z+2	
8E9B	91 28	116	STA (#28),Y	
8E9D	AD 15 03	117	LDA CHAR	;PUT RETURN CHR. IN A
8EA0	60	118	RTS	
8EA1	AD 07 03	119	STARTC1 LDA KEY+1	
8EA4	CD F7 91	120	CMP LKEY+1	
8EA7	D0 18	121	BNE STKEY	
8EA9	AD 06 03	122	LDA KEY	
8EAC	CD F6 91	123	CMP LKEY	
8EAF	D0 10	124	BNE STKEY	
8EB1	CE 17 03	125	DEC N+1	
8EB4	D0 08	126	BNE STKEY	
8EB6	AD 11 03	127	LDA RDFL	;GET A KDB ?
8EB9	D0 C9	128	BNE STARTC2	;NO RETURN
8EBB	20 7D 92	129	JSR GETKDB	
8EBE	4C 84 8E	130	JMP STARTC2	;RESTORE REG'S & RET.
8EC1	AD 06 03	131	STKEY LDA KEY	
8EC4	8D F6 91	132	STA LKEY	
8EC7	AD 07 03	133	LDA KEY+1	
8ECA	8D F7 91	134	STA LKEY+1	
8ECD	4C 72 8E	135	JMP STARTC	
8ED0		136	*	
8ED0		137	***** READ RY,RX1,RX2	

8ED0		138	*		
8ED0	4C 75 8F	139	NONO2	JMP NONO	
8ED3	20 F8 91	140	READKEY	JSR KPON	:SENSOR ON?
8ED6	90 F8	141		BCC NONO2	:NO
8ED8	A9 00	142		LDA #00	
8EDA	8D DC 91	143		STA YCNT	
8EDD	8D DB 91	144		STA XCNT	
8EE0	A9 FF	145		LDA #FF	
8EE2	A0 08	146		LDY #08	
8EE4	99 DF 91	147	LOOP3	STA RESTABXL,Y	:SET TABLES TO HIGH VALUE
8EE7	88	148		DEY	
8EE8	D0 FA	149		BNE LOOP3	
8EEA	AE F4 91	150	READX1	LDX X1PDL	:PREPARE TO READ LEFT X-STRIP
8EED	20 DB 90	151		JSR RES.READ	
8EF0	F0 DE	152		BEQ NONO2	:BAD READ
8EF2	20 1C 91	153		JSR SAVEX	
8EF5	A2 00	154		LDX #00	
8EF7	20 38 91	155		JSR CHECKXRD	:IS THERE 2 EQUAL READS ?
8EFA	F0 EE	156		BEQ READX1	:NO, READ AGAIN
8EFC	AD 03 8E	157		LDA LASTREAD	
8EFF	CD 23 94	158		CMP CALTAB+20	
8F02	90 10	159		BLT ITSX1	
8F04	AD 03 8E	160		LDA LASTREAD	
8F07	CD 23 94	161		CMP CALTAB+20	
8F0A	D0 11	162		BNE READX2	
8F0C	AD 04 8E	163		LDA LASTREAD+1	
8F0F	CD 22 94	164		CMP CALTAB+19	:YES,HIGHER THAN RANK 22 ?
8F12	B0 09	165		BGE READX2	:YES,ITS THE RIGHT STRIP
8F14	AD F4 91	166	ITSX1	LDA X1PDL	:NO,SET STRIP FLAG=0
8F17	8D DE 91	167		STA STFL	
8F1A	4C 23 8F	168		JMP READST	:GO READ
8F1D	AD F5 91	169	READX2	LDA X2PDL	:SET STRIP FLAG=1
8F20	8D DE 91	170		STA STFL	
8F23	A9 FF	171	READST	LDA #FF	
8F25	A0 04	172		LDY #04	
8F27	99 DF 91	173	LOOP4	STA RESTABXL,Y	
8F2A	88	174		DEY	
8F2B	D0 FA	175		BNE LOOP4	
8F2D	A9 00	176		LDA #00	
8F2F	8D DB 91	177		STA XCNT	
8F32	AE F3 91	178		LDX YPDL	:PREPARE TO READ Y
8F35	20 DB 90	179		JSR RES.READ	
8F38	F0 96	180		BEQ NONO2	:BAD READ START OVER
8F3A	20 0F 91	181		JSR SAVEY	
8F3D	20 BE 91	182		JSR STOREY	
8F40	AE DE 91	183		LDX STFL	:PREPARE TO READ X-STRIP
8F43	20 DB 90	184		JSR RES.READ	
8F46	F0 88	185		BEQ NONO2	:BAD READ START OVER
8F48	20 1C 91	186		JSR SAVEX	
8F4B	20 64 91	187		JSR STOREX	
8F4E	AE F3 91	188	READAG	LDX YPDL	:PREPARE TO READ Y AGAIN
8F51	20 DB 90	189		JSR RES.READ	
8F54	F0 1F	190		BEQ NONO	:BAD READ START OVER
8F56	20 0F 91	191		JSR SAVEY	
8F59	AE DE 91	192		LDX STFL	:PREPARE TO READ X
8F5C	20 DB 90	193		JSR RES.READ	
8F5F	F0 14	194		BEQ NONO	:BAD READING
8F61	20 1C 91	195		JSR SAVEX	

8F64	A2	00	196		LDX ##00	
8F66	20	7F	197		JSR CHECKYRD	:ARE THESE READINGS OK ?
8F69	D0	1A	198		BNE ACCEPT	:YES ACCEPT
8F6B	AD	00	199		LDA RTFL	
8F6E	C9	55	200		CMP ##55	:IS TABLE FULL?
8F70	D0	DC	201		BNE READAG	:NO READ
8F72	4C	D3	202		JMP READKEY	:DON'T ACCEPT
8F75	A9	00	203	NON0	LDA ##00	:BAD READING
8F77	8D	D7	204		STA FLAG	:RESET 1 SEC.WAIT
8F7A	A9	40	205		LDA ##40	
8F7C	8D	D6	206		STA CCOUNT	
8F7F	A9	FF	207		LDA ##FF	:RETURN FLAG
8F81	8D	00	208		STA RTFL	
8F84	60		209		RTS	
8F85	A9	00	210	ACCEPT	LDA ##00	
8F87	8D	00	211		STA RTFL	
8F8A	AD	E8	212		LDA LASTX+1	
8F8D	8D	01	213		STA RY	
8F90	CD	0F	214		CMP CALTAB	:HIGHER THAN RANK 18 ?
8F93	B0	E0	215		BGE NON0	:YES REJECT
8F95	AD	DE	216		LDA STFL	:WHICH X-STRIP WAS IT ?
8F98	D0	15	217		BNE RACCEPT	:RIGHT STRIP
8F9A	AD	E9	218		LDA LASTX	
8F9D	CD	23	219		CMP CALTAB+20	
8FA0	90	1F	220		BLT LACCEPT	
8FA2	D0	D1	221		BNE NON0	
8FA4	AD	EA	222		LDA LASTX+1	:LEFT STRIP
8FA7	CD	22	223		CMP CALTAB+19	:GREATER THAN RANK 22 ?
8FAA	B0	C9	224		BGE NON0	:YES REJECT READING
8FAC	4C	C1	225		JMP LACCEPT	
8FAF	AD	E9	226	RACCEPT	LDA LASTX	
8FB2	09	16	227		ORA ##16	
8FB4	CD	7D	228		CMP CALTAB+110	
8FB7	D0	BC	229		BNE NON0	
8FB9	AD	EA	230		LDA LASTX+1	:LEFT STRIP
8FBC	CD	7C	231		CMP CALTAB+109	:IS IT LARGER THAN RANK 44 ?
8FBF	B0	B4	232		BGE NON0	:YES,SO START OVER
8FC1	20	D4	233	LACCEPT	JSR ACCEPT2	:GET SHIFT STATUS
8FC4	AD	DE	234		LDA STFL	:WAS READING ON LEFT STRIP ?
8FC7	F0	08	235		BEQ COMP	:YES ,COMPUTE ROW & COL.
8FC9	A9	16	236		LDA ##16	:NO,ADD #16 TO DESIGNATE
8FCB	0D	E9	237		ORA LASTX	:RIGHT STRIP
8FCE	8D	03	238		STA RX+1	
8FD1	4C	EE	239	COMP	JMP COMPIJ	
8FD4	AD	EA	240	ACCEPT2	LDA LASTX+1	:GET X READING
8FD7	8D	02	241		STA RX	
8FDA	AD	E9	242		LDA LASTX	:GET HIGH ORDER BYTE
8FDD	8D	03	243		STA RX+1	
8FE0	A9	00	244		LDA ##00	:NO SHIFT
8FE2	8D	08	245		STA SHIFT	
8FE5	2C	62	246		BIT S2SWITCH	:TYPEWRITER SHIFT?
8FE8	10	03	247		BPL NOS3	:NO
8FEA	EE	08	248		INC SHIFT	:PUT 1 IN SHIFT FLAG
8FED	60		249	NOS3	RTS	
8FEE			250	*		
8FEE			251		***** COMPUTE ROW & COLUMN(I&J)	
8FEE			252	*		
8FEE	20	B8	253	COMPIJ	JSR HIGHRD	:STORE HIGHEST VALUE TO COMP.

8FF1 A2 01	254	LDX ##01	
8FF3 A0 01	255	LDY ##01	
8FF5 20 CA 90	256	JSR GETADR	:GET ADDRESS OF CALIB. TABLE
8FF8 A0 11	257	LDY #17	:PREPARE FOR HIGHEST VALUE RY
8FFA AD 01 03	258	NEXTCTB LDA RY	
8FFD 38	259	SEC	
8FFE F9 11 11	260	TBL2 SBC #1111,Y	:SUBTR.VALUE CALTAB AGAINST RY
9001 B0 03	261	BCS TEST	:RESULT IS POSI. SO CMP
9003 20 C1 90	262	JSR ABSVALU	:CONVERTS TO POSITIVE #
9006 CD EB 91	263	TEST CMP LAST	:IS THIS VALUE CALTAB CLOSER?
9009 B0 06	264	BGE ACCY	:NO SAVE PREVIOUS
900B 8D EB 91	265	STA LAST	:YES SO SAVE THIS VALUE
900E 88	266	DEY	:CHECK NEXT VAUE CALTAB
900F 10 E9	267	BPL NEXTCTB	:GO READ TABLE AGAIN
9011 C8	268	ACCY INY	:RESTORES Y TO LAST VALUE
9012 98	269	TYA	
9013 8D 04 03	270	STA ROW	:STORES IT AS CORRECT ROW #
9016 8D 48 92	271	STA M2	:LOC. USED BY MULT ROUTINE
9019 A9 2C	272	LDA #44	:# OF COLUMNS TO BE ADDED BY
901B 8D 46 92	273	STA M1	:LOC. USED BY MULT.
901E 20 48 92	274	JSR MULT	:CALC. KEY # ASSUM.COL=0
9021	275	*	
9021	276	***** FIND COLUMN #	
9021	277	*	
9021 20 B8 90	278	JSR HIGHRD	:SET LAST READING TO HIGH #
9024 A2 00	279	LDX ##00	:PREPARE TO FIND COLUMN
9026 18	280	CLC	
9027 A9 15	281	LDA #21	:PREPARE TO READ CALIBTAB OF X
9029 6D 05 8E	282	ADC ADCALIB	:LOW ORDER BYTE(LOB) OF X TAB.
902C 8D 4E 90	283	STA TBL3+1	
902F A9 00	284	LDA ##00	:HIGH ORDER BYTE(HOB) FOR ADDR
9031 6D 06 8E	285	ADC ADCALIB+1	:OF LOB OF CALIB. X TABLE
9034 8D 4F 90	286	STA TBL3+2	
9037 AD 4E 90	287	LDA TBL3+1	
903A 69 01	288	ADC ##01	:ADD 1 TO GET LOB OF HOB ADDR.
903C 8D 57 90	289	STA TBL4+1	:FOR X'S CALIB.LOC.
903F A9 00	290	LDA ##00	
9041 6D 4F 90	291	ADC TBL3+2	:HOB OF HOB ADDR. FOR X TBL.
9044 8D 58 90	292	STA TBL4+2	
9047 A0 56	293	LDY #86	:CHECK HIGHEST RANK FIRST
9049 AD 02 03	294	RESX LDA RX	
904C 38	295	SEC	
904D F9 11 11	296	TBL3 SBC #1111,Y	:SUBTRACT RX VALUE FROM CALTAB
9050 8D ED 91	297	STA XLAST	:SAVE
9053 AD 03 03	298	LDA RX+1	
9056 F9 11 11	299	TBL4 SBC #1111,Y	:SUBTRACT HOB'S
9059 8D EE 91	300	STA XLAST+1	
905C B0 1F	301	BCS XTEST	:POS. READING?,YES BRANCH
905E AD EE 91	302	LDA XLAST+1	
9061 20 C1 90	303	JSR ABSVALU	:NO.CHANGE TO POSITIVE VALUE
9064 8D EE 91	304	STA XLAST+1	
9067 AD ED 91	305	LDA XLAST	
906A 20 C1 90	306	JSR ABSVALU	:CHANGE TO POSITIVE VALUE
906D 8D ED 91	307	STA XLAST	
9070 CE EE 91	308	DEC XLAST+1	:DELETE CARRY GENERATED
9073 A9 00	309	LDA ##00	
9075 CD ED 91	310	CMP XLAST	:IS XLAST = 0
9078 D0 03	311	BNE XTEST	:NO



907A	EE	EE	91	312		INC XLAST+1		:YES,RESET CARRY
907D	AD	EE	91	313	XTEST	LDA XLAST+1		
9080	CD	EC	91	314		CMF LAST+1		:IS NEW READ CLOSER THAN LAST
9083	B0	03		315		BGE ACCX1		:NO,ACCEPT PREVIOUS
9085	4C	90	90	316		JMP DNTACX		:YES,CHECK NEXT CALTAB VALUE
9088	AD	ED	91	317	ACCX1	LDA XLAST		
908B	CD	EB	91	318		CMF LAST		:IS NEW READ LOB CLOSER
908E	B0	11		319		BGE ACCX		:NO,ACCEPT PREVIOUS
9090	AD	ED	91	320	DNTACX	LDA XLAST		:YES,CHECK NEXT CALTAB VALUE
9093	8D	EB	91	321		STA LAST		:MAKE NEW CLOSER READ LAST
9096	AD	EE	91	322		LDA XLAST+1		
9099	8D	EC	91	323		STA LAST+1		:MAKE NEW CLOSER READ LAST
909C	88			324		DEY		:GO BACK TWO BYTES TO RESET
909D	88			325		DEY		:TO NEXT CALTAB LOC.
909E	4C	49	90	326		JMP RESX		:SEE IF NEXT CALTAB IS CLOSER
90A1	C8			327	ACCX	INY		:RESTORE Y TO PREVIOUS
90A2	C8			328		INY		:CLOSEST VALUE
90A3	98			329		TYA		
90A4	4A			330		LSR		
90A5	8D	05	03	331		STA COLUMN		:SAVE A AS CORRECT COLUMN #
90A8	18			332		CLC		
90A9	6D	49	92	333		ADC M3		:ADD COL. # TO M3 TO GET KEY#
90AC	8D	06	03	334		STA KEY		
90AF	A9	00		335		LDA ##00		
90B1	6D	4A	92	336		ADC M3+1		:SAVE HOB OF KEY #
90B4	8D	07	03	337		STA KEY+1		
90B7	60			338		RTS		
90B8				339	*			
90B8				340		***** HIGH VALUE TO COMP.AGAINST		
90B8				341	*			
90B8	A9	FF		342	HIGHRD	LDA ##FF		
90BA	8D	EB	91	343		STA LAST		
90BD	8D	EC	91	344		STA LAST+1		
90C0	60			345		RTS		
90C1	8D	C8	90	346	ABSVALU	STA ABSVALU1+1		:STORES READING DIFF.IN ACCUM
90C4	38			347		SEC		
90C5	A9	00		348		LDA ##00		
90C7	E9	11		349	ABSVALU1	SBC ##11		:RETURNS RES.DIFF.AS A POS.#
90C9	60			350		RTS		
90CA	98			351	GETADR	TYA		
90CB	18			352		CLC		
90CC	6D	05	8E	353		ADC ADCALIB		:ADDS 1 TO GET LOB OF Y CALTAB
90CF	8D	FF	8F	354		STA TBL2+1		:LOB
90D2	A9	00		355		LDA ##00		
90D4	6D	06	8E	356		ADC ADCALIB+1		:GETS HOB OF Y CALTAB
90D7	8D	00	90	357		STA TBL2+2		:HOB
90DA	60			358		RTS		
90DE				359	*			
90DE				360		***** READ RESISTANCE STRIP # X		
90DE				361	*			
90DB	8E	DD	91	362	RES.READ	STX XREG		:READ WHICH STRIP?(00,01,02)
90DE	8A			363		TXA		
90DF	18			364		CLC		
90E0	69	64		365		ADC ##64		:ADD STRIP # TO 64
90E2	8D	F5	90	366		STA PDL+1		:TO GET PADDLE PORT # TO READ
90E5	AD	18	03	367	STRD	LDA WT		
90E8	20	A8	FC	368		JSR #FCAB		:WAIT BEFORE READING
90EB	A2	00		369		LDX ##00		

90ED AD 70 C0	370	LDA #C070	:TRIGGER PADL TO STRT RES.READ
90F0 A0 00	371	LDY ##00	
90F2 EA	372	NOP	:STALL BEFORE READING
90F3 EA	373	NOP	
90F4 AD 11 C0	374	PDL LDA #C011	:READ FADDLE
90F7 10 0B	375	BFL SAVRD	:IF OFF SAVE R READING
90F9 C8	376	INY	:IF STILL ON INC.COUNT
90FA D0 F8	377	BNE FDL	:GO COUNT AGAIN
90FC E8	378	INX	:INC HOB
90FD E0 02	379	CPX ##02	
90FF D0 F3	380	BNE FDL	:GO READ
9101 A9 00	381	DNAR LDA ##00	:DON'T ACCEPT READING
9103 60	382	RTS	:STORE 0 AS ERROR FLAG & RET.
9104 8C 04 BE	383	SAVRD STY LASTREAD+1	:STORE Y VALUE AS GOOD COUNT
9107 8A	384	TXA	
9108 38	385	SEC	
9109 8D 03 BE	386	STA LASTREAD	:SET HOB
910C A9 FF	387	LDA ##FF	
910E 60	388	RTS	
910F AD 03 BE	389	SAVEY LDA LASTREAD	
9112 8D E7 91	390	STA LASTY	
9115 AD 04 BE	391	LDA LASTREAD+1	
9118 8D EB 91	392	STA LASTY+1	
911B 60	393	RTS	
911C AD 03 BE	394	SAVEX LDA LASTREAD	
911F 8D E9 91	395	STA LASTX	
9122 AD 04 BE	396	LDA LASTREAD+1	
9125 8D EA 91	397	STA LASTX+1	
9128 60	398	RTS	
9129	399	*	
9129	400	***** TRAP	
9129	401	*	
9129 20 A4 93	402	INIT JSR CDF	:SET TRAPS IN USER KEYIN
912C EF 91	403	ADR FSTART	
912E 55 AA	404	HEX 55AA	
9130 20 A4 93	405	JSR CDF	
9133 F1 91	406	ADR DOS,	
9135 38 00	407	HEX 3800	
9137 60	408	RTS	
9138	409	*	
9138	410	***** CHECK X READINGS	
9138	411	*	
9138 BD E1 91	412	CHECKXRD LDA RESTABXH,X	:GET LAST READINGS
913B CD E9 91	413	CMP LASTX	:IS IT THE SAME AS LSTRD?
913E D0 0B	414	BNE CHECKX	:NO GET ANOTHER READING
9140 BD DF 91	415	LDA RESTABXL,X	:YES CHECK LOB
9143 CD EA 91	416	CMP LASTX+1	:IS IT EQUAL
9146 D0 03	417	BNE CHECKX	:NO
9148 4C 76 91	418	JMP ACC	:ACCEPT
914B E8	419	CHECKX INX	
914C E0 02	420	CPX ##02	:HAVE WE CHECKED BOTH POSS.
914E D0 E8	421	BNE CHECKXRD	:NO.CHECK NEXT
9150 EE DB 91	422	NOACC INC XCNT	
9153 AC DB 91	423	LDY XCNT	
9156 C0 02	424	CPY ##02	:IS TABLE FULL ?
9158 D0 0A	425	BNE STOREX	:NO,STORE THIS VALUE
915A A9 55	426	LDA #55	
915C 8D 00 03	427	STA RTFL	

915F A0 00	428	LDY ##00	;YES,START WRITING OVER OLDEST
9161 8C DB 91	429	STY XCNT	
9164 AD E9 91	430	LDA LASTX	;STORE NEW READ IN TBL.
9167 AC DB 91	431	LDY XCNT	
916A 99 E1 91	432	STA RESTABXH,Y	;PLACE # IN TBL
916D AD EA 91	433	LDA LASTX+1	;STORE LOB
9170 99 DF 91	434	STA RESTABXL,Y	
9173 4C 01 91	435	JMP DNAR	;DON'T ACCEPT
9176 AE DD 91	436	ACC LDX XREG	
9179 A9 FF	437	LDA ##FF	
917B 8D 00 03	438	STA RTFL	
917E 60	439	RTS	
917F	440	*	
917F	441	***** CHECK Y READINGS	
917F	442	*	
917F BD E5 91	443	CHECKYRD LDA RESTABYH,X	;GET LAST READINGS
9182 CD E7 91	444	CMP LASTY	;IS IT THE SAME AS LASTY?
9185 D0 1A	445	BNE CHECKY	;NO GET ANOTHER READING
9187 BD E3 91	446	LDA RESTABYL,X	;YES CHECK LOB
918A CD E8 91	447	CMP LASTY+1	;IS IT EQUAL
918D D0 12	448	BNE CHECKY	;NO GET NEXT
918F BD E1 91	449	LDA RESTABXH,X	;IS THE SAME X EQUAL
9192 CD E9 91	450	CMP LASTX	
9195 D0 39	451	BNE NOACC1	;NO STORE VALUES + READ
9197 BD DF 91	452	LDA RESTABXL,X	;YES IS LOB EQUAL
919A CD EA 91	453	CMP LASTX+1	
919D F0 D7	454	BEQ ACC	;YES ACCEPT THESE READINGS
919F D0 2F	455	BNE NOACC1	
91A1 E8	456	CHECKY INX	
91A2 E0 02	457	CPX ##02	;HAVE WE CHECKED BOTH POSS.
91A4 D0 D9	458	BNE CHECKYRD	;NO,CHECK NEXT
91A6 4C D0 91	459	JMP NOACC1	
91A9 AC DC 91	460	NOACC2 LDY YCNT	
91AC 8C DB 91	461	STY XCNT	
91AF EE DC 91	462	INC YCNT	
91B2 AC DC 91	463	LDY YCNT	
91B5 C0 02	464	CPY ##02	;IS THE TABLE FULL ?
91B7 D0 05	465	BNE STOREY	;NO,STORE THIS VALUE
91B9 A0 00	466	LDY ##00	;YES,START WRITING OVER OLDEST
91BB 8C DC 91	467	STY YCNT	
91BE AD E7 91	468	STOREY LDA LASTY	;STORE NEW READ IN TBL.
91C1 AC DC 91	469	LDY YCNT	
91C4 99 E5 91	470	STA RESTABYH,Y	;PLACE # IN TBL
91C7 AD E8 91	471	LDA LASTY+1	
91CA 99 E3 91	472	STA RESTABYL,Y	
91CD 4C 01 91	473	JMP DNAR	;STORE X READINGS
91D0 20 A9 91	474	NOACC1 JSR NOACC2	
91D3 4C 50 91	475	JMP NOACC	
91D6	476	*	
91D6	477	***** DATA	
91D6	478	*	
91D6 00	479	CCOUNT HEX 00	
91D7 00	480	FLAG HEX 00	
91D8 00 00 00	481	Z HEX 000000	
91DB 00	482	XCNT HEX 00	
91DC 00	483	YCNT HEX 00	
91DD 00	484	XREG HEX 00	
91DE 00	485	STFL HEX 00	

91DF 00 00	486	RESTABXL	HEX 0000	
91E1 00 00	487	RESTABXH	HEX 0000	
91E3 00 00	488	RESTABYL	HEX 0000	
91E5 00 00	489	RESTABYH	HEX 0000	
91E7 00 00	490	LASTY	HEX 0000	
91E9 00 00	491	LASTX	HEX 0000	
91EB 00 00	492	LAST	HEX 0000	
91ED 00 00	493	XLAST	HEX 0000	
91EF 07 BE	494	FSTART	ADR START	
91F1 81 9E	495	DOS	HEX 819E	
91F3 02	496	YFDL	HEX 02	
91F4 00	497	X1PDL	HEX 00	
91F5 01	498	X2PDL	HEX 01	
91F6 00 00	499	LKEY	HEX 0000	
91F8	500	*		
91F8	501	*		
91F8	502	*****	KEYPORT ON ?	
91F8	503	*		
91F8 48	504	KPON	PHA	:SAVE REGS
91F9 98	505		TYA	
91FA 48	506		PHA	
91FB 8A	507		TXA	
91FC 48	508		PHA	
91FD A9 03	509		LDA #03	
91FF 20 A8 FC	510		JSR \$FCAB	
9202 AE F3 91	511		LDX YPDL	:PREPARE TO READ Y STRIP
9205 20 DB 90	512		JSR RES.READ	
9208 F0 14	513		BEQ KPOFF	:BAD READ
920A AD 04 BE	514		LDA LASTREAD+1	
920D CD 0F 94	515		CMP CALTAB	:HIGHER THAN RANK 18 ?
9210 B0 0C	516		BGE KPOFF	:YES,KP IS OFF
9212 A9 01	517		LDA #01	
9214 8D 00 03	518		STA RTFL	
9217 38	519		SEC	:NO,SET CARRY AS FLAG KP ON
9218 68	520	KPRTS	FLA	
9219 AA	521		TAX	:RESTORE REGS
921A 68	522		FLA	
921B A8	523		TAY	
921C 68	524		PLA	
921D 60	525		RTS	:RTS,WITH CARRY STATUS FLAG
921E 2C 00 C0	526	KPOFF	BIT \$C000	:KEYBOARD ON ?
9221 10 09	527		BFL KPOFF2	
9223 A9 02	528		LDA #02	:YES,SET FLAG=2
9225 8D 00 03	529		STA RTFL	
9228 18	530		CLC	:CLEAR CARRY AS FLAG KP OFF
9229 4C 18 92	531		JMP KPRTS	
922C A9 00	532	KPOFF2	LDA #00	:NO,SET FLAG=0
922E 8D 00 03	533		STA RTFL	:MEANING NO KEY IS PRESSED
9231 8D 07 91	534		STA FLAG	
9234 A9 40	535		LDA #40	
9236 8D D6 91	536		STA CCOUNT	
9239 8D F7 91	537		STA LKEY+1	
923C AD 16 03	538		LDA N	
923F 8D 17 03	539		STA N+1	
9242 18	540		CLC	:KP OFF SO CLEAR CARRY
9243 4C 18 92	541		JMP KPRTS	:RESTORE REGISTERS & RET.
9246	542	*		
9246	543	*****	MULTIPLICATION ROUTINE: M1*M2=M3	

9246		544	*		
9246	00 00	545	M1	HEX 0000	:1ST PARAMETER(1 BYTE)
9248	00	546	M2	HEX 00	:2ND PARAMETER(1 BYTE)
9249	00 00	547	M3	HEX 0000	:RESULT (2 BYTES)
924B	A9 00	548	MULT	LDA #00	
924D	8D 49 92	549		STA M3	
9250	8D 47 92	550		STA M1+1	
9253	8D 4A 92	551		STA M3+1	
9256	AD 48 92	552	MLO1	LDA M2	:CHECK M2
9259	F0 21	553		BEQ RTSM	:IF ZERO,ALL BITS CHECKED RET
925B	4E 48 92	554		LSR M2	:SHIFT,CHECK FIRST BIT
925E	90 13	555		BCC MULT2	:IF ONE DON'T ADD
9260	AD 46 92	556		LDA M1	:IF 1ST BIT IS ZERO
9263	18	557		CLC	
9264	6D 49 92	558		ADC M3	:ADD LOB TO LAST VALUE OF M3
9267	8D 49 92	559		STA M3	:TO GET LATEST KEY VALUE
926A	AD 47 92	560		LDA M1+1	
926D	6D 4A 92	561		ADC M3+1	:ADD HOB'S
9270	8D 4A 92	562		STA M3+1	
9273	0E 46 92	563	MULT2	ASL M1	:MULT.44*2 TO GET NEXT POSS.#
9276	2E 47 92	564		ROL M1+1	:Y VALUE
9279	4C 56 92	565		JMP MLO1	:GO CHECK NEXT BIT
927C		566		LST	
927C	60	567	RTSM	RTS	:RETURN WITH KEY # IN M3&M3+1
927D		568	*		
927D		569	*****	GET KEY DATA BLOCK (KDB)	
927D		570	*		
927D	20 A4 93	571	GETKDB	JSR COP	
9280	06 03	572		ADR KEY	
9282	0A 94	573		ADR KEY2	
9284	0E 0A 94	574		ASL KEY2	:KEY2*2 TO GET POINTER LOC
9287	2E 0B 94	575		ROL KEY2+1	:MULT.HOB
928A	A0 03	576		LDY #03	
928C	A9 00	577		LDA #00	
928E	99 09 03	578	LOOP1	STA ADKDB,Y	:CLEARS LOC.
9291	88	579		DEY	
9292	88	580		DEY	
9293	10 F9	581		BPL LOOP1	
9295	20 3D 93	582		JSR ADD	:KEY2+TABD=TABL ENTRY POINT
9298	0A 94	583		ADR KEY2	
929A	0D 03	584		ADR TABD	
929C	A3 92	585		ADR POINTER	:POINTER IN TABLE
929E	20 DD 93	586		JSR ADDA	:GET VALUE OF POINTER
92A1	00 00	587		HEX 0000	
92A3	11 11	588	POINTER	HEX 1111	
92A5	BB 92	589		ADR DUMM+1	
92A7	A9 FF	590		LDA #FFF	:VALUE=FFF,UNASIGNED KEY
92A9	CD BC 92	591		CMP DUMM+2	:IS HOB FFF
92AC	F0 47	592		BEQ RETDNA	:YES.RETURN ERROR
92AE	20 3D 93	593		JSR ADD	:ADDS VALUES IN NEXT 2 BYTES
92B1	0D 03	594		ADR TABD	:TO THE NEXT 2 BYTES & STORES
92B3	BB 92	595		ADR DUMM+1	:RESULT IN THE 3RD 2 BYTES
92B5	BB 92	596		ADR DUMM+1	:POINTS TO KDB
92B7	AC 0B 03	597		LDY SHIFT	:SHIFT ?
92BA	AD 11 11	598	DUMM	LDA #1111	:KDB ADDR.1ST BYTE=LENGTH
92BD	8D C6 92	599		STA LENTH	:SAVE THIS VALUE
92C0	88	600		DEY	:SHIFT=0
92C1	30 11	601		BNJ OUTRET	:YES,ACCEPT THIS KDB

92C3	20	DD	93	602		JSR ADDA		:NO.ADD LENGTH TO
92C6	00	00		603	LENTH	HEX 0000		:KDB ADDR. TO GET NEXT
92C8	BB	92		604		ADR DUMM+1		:ENTRY POINT OF KDB
92CA	BB	92		605		ADR DUMM+1		:WITH A SHIFT
92CC	20	EF	93	606		JSR INC		:INC.THE NEXT 2 BYTES
92CF	BB	92		607		ADR DUMM+1		
92D1	4C	BA	92	608		JMP DUMM		:CHECK SHIFT AGAIN
92D4	AD	C6	92	609	OUTRET	LDA LENTH		:IS LENGTH=0
92D7	F0	1C		610		BED RETDNA		:YES.RETURN ERROR
92D9	8D	0C	03	611		STA LEN		:NO.SAVE LEN.
92DC	20	A4	93	612		JSR COP		:MOVES CONT.OF DUMM+1 TO ADKDB
92DF	BB	92		613		ADR DUMM+1		
92E1	09	03		614		ADR ADKDB		
92E3	20	DD	93	615		JSR ADDA		:ADDS 1 TO KDB ADDR.
92E6	01	00		616		HEX 0100		:TO GET FUN. # ADDR.
92E8	BB	92		617		ADR DUMM+1		
92EA	ED	92		618		ADR GETFN+1		:STORES RESULT HERE
92EC	AD	11	11	619	GETFN	LDA #1111		:POINTS TO FUNTCION #
92EF	8D	0B	03	620		STA FUN		:SAVES FUNCTION #
92F2	4C	FB	92	621		JMP MOVE2		
92F5	A9	03		622	RETDNA	LDA #03		
92F7	8D	00	03	623		STA RTFL		
92FA	60			624		RTS		
92FB				625	*			
92FB				626		***** MOVE KDB TO #201		
92FB				627	*			
92FB	A9	01		628	MOVE2	LDA #01		:LOB OF DESTINATION
92FD	85	42		629		STA #42		
92FF	A9	02		630		LDA #02		:HOB OF DESTINATION
9301	85	43		631		STA #43		
9303	20	DD	93	632		JSR ADDA		:ADD 2 TO GET FAST
9306	02	00		633		HEX 0200		:FUN & LENGHT
9308	09	03		634		ADR ADKDB		
930A	0F	93		635		ADR TEMPKDB		
930C	4C	11	93	636		JMP OUTRET3		
930F	11	11		637	TEMPKDB	HEX 1111		
9311	AD	0F	93	638	OUTRET3	LDA TEMPKDB		:GET LOB OF START SOURCE ADDR
9314	85	3C		639		STA #3C		
9316	AD	10	93	640		LDA TEMPKDB+1		:GET HOB OF START SOURCE ADDR
9319	85	3D		641		STA #3D		
931B	18			642		CLC		
931C	AD	0C	03	643		LDA LEN		:ADD LENGTH OF STRING TO GET
931F	6D	0F	93	644		ADC TEMPKDB		:ENDING ADDR.FOR MOVE
9322	8D	0F	93	645		STA TEMPKDB		
9325	A9	00		646		LDA #00		
9327	6D	10	93	647		ADC TEMPKDB+1		
932A	8D	10	93	648		STA TEMPKDB+1		
932D	AD	0F	93	649		LDA TEMPKDB		:GET LOB OF END SOURCE ADDR
9330	85	3E		650		STA #3E		
9332	AD	10	93	651		LDA TEMPKDB+1		:GET HOB OF END SOURCE ADDR
9335	85	3F		652		STA #3F		
9337	A0	00		653		LDY #00		
9339	20	2C	FE	654		JSR \$FE2C		:MONITOR MOVE
933C	60			655		RTS		:RET.W/GOOD ADKDB,FUN,LEN
933D				656	*			
933D				657		*****		
933D				658	*	UTILITY SUBS THAT		
933D				659	*	ADD BY MANIPULATION OF		

933D		660	*	OF THE STACK	
933D		661	*****		
933D		662	*	SUB: ADD	
933D		663	*	FORMAT: JSR ADD	
933D		664	*	ADR VAR1	
933D		665	*	ADR VAR2	
933D		666	*	RESULT: ADR VAR3	
933D		667	*		
933D		668	*	ADDS VAR1 TO VAR2 RESULT IN VAR3	
933D		669	*****		
933D		670	*		
933D	A9 06	671	ADD	LDA #06	:STACK TO RET.6 BYTES LATER
933F	20 6C 93	672		JSR UPST	:ADJUST STACK & GET ITS ADDR
9342	A0 00	673		LDY #00	
9344	20 D3 93	674		JSR AD1	:GET ADDR.OF DATA TO ADD
9347	20 BB 93	675		JSR FBBUF	:SAVE THE DATA IN THIS ADDR.
934A	A0 02	676	ADD10	LDY #02	
934C	20 D3 93	677		JSR AD1	:GET NEXT 2 BYTES ADDR.
934F	A0 00	678		LDY #00	
9351	18	679		CLC	
9352	20 63 93	680		JSR ADD2	:ADD THE 2 LOB'S
9355	C8	681		INX	
9356	20 63 93	682		JSR ADD2	:ADD THE 2 HOB'S
9359	A0 04	683		LDY #04	
935B	20 D3 93	684		JSR AD1	:GET ADDR.TO STORE DATA IN
935E	20 C6 93	685		JSR BUFFB	:STORE DATA
9361	30 34	686		BMI RETURN	:RESTORE X & Y RET.
9363	B1 FB	687	ADD2	LDA (#FB),Y	:GETS DATA IN LAST READ
9365	79 D1 93	688		ADC BUF,Y	:& ADDS IT TO PREVIOUS READ
9368	99 D1 93	689		STA BUF,Y	:SAVES IT IN BUFFER
936B	60	690		RTS	
936C		691	*		
936C		692	*****		
936C		693	*		
936C		694	*	SAVES ADDR OF 2ND RTS IN	
936C		695	*	#F9 & #FA THEN CHANGES 2ND	
936C		696	*	RTS TO + THE # OF BYTES IN ACCUM.	
936C		697	*		
936C		698	*****		
936C		699	*		
936C	8C 0D 94	700	UPST	STY YX	:PRESERVE Y VALUE
936F	8E 0E 94	701		STX YX+01	:PRESERVE X VALUE
9372	BA	702		TSX	:MOVE STACK TO X
9373	8D 0C 94	703		STA MOVE	:# OF BYTES STACK IS MOVED
9376	E8	704		INX	:ADD 3 TO STACK TO GET
9377	E8	705		INX	:TO THE 2ND RETURN ADDR.
9378	E8	706		INX	
9379	BD 00 01	707		LDA #100,X	:GET LOB OF CALLING ROUTINE
937C	85 F9	708		STA #F9	:SAVE IT ON UNUSED 0 PAGE
937E	18	709		CLC	
937F	6D 0C 94	710		ADC MOVE	:ADD STACK + A TO GET NEW RET.
9382	9D 00 01	711		STA #100,X	:STORES IT BACK IN STACK
9385	E8	712		INX	
9386	BD 00 01	713		LDA #100,X	:GET HOB
9389	85 FA	714		STA #FA	:SAVE HOB,UNUSED 0 PAGE
938B	69 00	715		ADC #00	
938D	9D 00 01	716		STA #100,X	:RESORE TO STACK
9390	E6 F9	717		INC #F9	:ADD 1 TO GET PROPER ADDR.

9392	D0 02	718	BNE RTS1	:IS LOB=0 ?
9394	E6 FA	719	INC #FA	:YES, INC. HOB
9396	60	720	RTS1 RTS	:NO, RETURN
9397	A0 00	721	RETURN LDY ##00	
9399	B1 FB	722	LDA (#FB),Y	
939B	AC 0D 94	723	LDY YX	:RESTORES Y BEFORE RET.
939E	AE 0E 94	724	LDX YX+#01	:RESTORES X BEFORE RET.
93A1	C9 00	725	CMP ##00	
93A3	60	726	RTS	
93A4		727	*	
93A4		728	*****	
93A4		729	*	
93A4		730	* SUB: COP	
93A4		731	* FORMAT: JSR COP	
93A4		732	* ADR VAR1	
93A4		733	* RESULT: ADR VAR2	
93A4		734	*	
93A4		735	* PUTS VAR1 IN VAR2	
93A4		736	*****	
93A4		737	*	
93A4	A9 04	738	COP LDA ##04	:# OF PLACES STACK IS MOVED
93A6	20 6C 93	739	JSR UPST	:ADJUST STACK & GET ITS ADDR
93A9	A0 00	740	LDY ##00	
93AB	20 D3 93	741	JSR AD1	:GET ADDR.OF DATA TO BE COPIED
93AE	20 BB 93	742	JSR FBUFF	:GET DATA IN THAT LOC.
93B1	A0 02	743	LDY ##02	
93B3	20 D3 93	744	JSR AD1	:GET ADDR.TO PUT DATA IN
93B6	20 C6 93	745	JSR FBUFF	:MOVE THE DATA TO THIS ADDR.
93B9	30 DC	746	BMI RETURN	:RESTORE REG.AND RETURNS
93BB		747	*	
93BB		748	*****	
93BB		749	*	
93BB		750	* LOADS BUFFER WITH DATA	
93BB		751	* IN ADDR. CONTAINED IN	
93BB		752	* LOC.(FB)	
93BB		753	*	
93BB		754	*****	
93BB		755	*	
93BB	A0 01	756	FBUFF LDY ##01	
93BD	B1 FB	757	COP1 LDA (#FB),Y	:GET DATA POINTED TO
93BF	99 D1 93	758	STA BUF,Y	:STORE #
93C2	88	759	DEY	:FINISHED
93C3	10 FB	760	BPL COP1	:NO,GET NEXT BYTE
93C5	60	761	RTS	:YES
93C6		762	*	
93C6		763	*****	
93C6		764	*	
93C6		765	* TAKES DATA IN THE BUFFER	
93C6		766	* STORES IT IN ADDR.POINTED	
93C6		767	* TO BY INDIRECT (#FB)	
93C6		768	*	
93C6		769	*****	
93C6		770	*	
93C6	A0 01	771	FBUFF LDY ##01	
93C8	B9 D1 93	772	LOOP2 LDA BUF,Y	:GET DATA
93CB	91 FB	773	STA (#FB),Y	:STORE IN NEW LOC.
93CD	88	774	DEY	:FINISHED
93CE	10 FB	775	BPL LOOP2	:NO GET NEXT BYTE



93D0 60	776	RTS	:YES
93D1	777	BUF DFS \$02	
93D3	778	*	
93D3	779	*****	
93D3	780	*	
93D3	781	* SUB: AD1	
93D3	782	*	
93D3	783	* GETS ADDR.POINTED TO BY UPST	
93D3	784	* READS THE ADDR IN THAT ADDR.	
93D3	785	* AND STORES IT IN (FB)&(FC)	
93D3	786	*****	
93D3	787	*	
93D3 B1 F9	788	AD1 LDA (\$F9),Y	:GET ADDR.POINTED TO BY (F9)
93D5 85 FB	789	STA \$FB	:SAVE LOB,UNUSED 0 PAGE
93D7 C8	790	INY	
93D8 B1 F9	791	LDA (\$F9),Y	:GET HOB POINTED TO BY (F9)
93DA 85 FC	792	STA \$FC	:SAVE HOB,UNUSED 0 PAGE
93DC 60	793	RTS	
93DD	794	*	
93DD	795	*****	
93DD	796	*	
93DD	797	* SUB: ADDA	
93DD	798	* FORMAT: JSR ADDA	
93DD	799	* HEX VAL1	
93DD	800	* ADR VAR2	
93DD	801	* RESULT: ADR VAR3	
93DD	802	*	
93DD	803	* ADDS VAL1 TO VAR2 RESULT VAR3	
93DD	804	*****	
93DD	805	*	
93DD A9 06	806	ADDA LDA #\$06	:# OF PLACES STACK IS MOVED
93DF 20 6C 93	807	JSR UPST	:ADJUST STACK & GET ITS ADDR.
93E2 A0 01	808	LDY #\$01	
93E4 B1 F9	809	XXADDA LDA (\$F9),Y	:GET 2 BYTES OF DATA
93E6 97 D1 93	810	STA BUF,Y	:PUT IN BUFFER
93E9 88	811	DEY	:FINISHED ?
93EA 10 FB	812	BPL XXADDA	:NO
93EC 4C 4A 93	813	JMP ADD10	:YES,GO GET NEXT DATA & ADD
93EF	814	*	
93EF	815	*****	
93EF	816	*	
93EF	817	* SUB: INC	
93EF	818	* FORMAT: JSR INC	
93EF	819	* ADR VAR1	
93EF	820	*	
93EF	821	* ADDS ONE TO VAR1 WITH	
93EF	822	* RESULT IN VAR1	
93EF	823	*****	
93EF	824	*	
93EF A9 02	825	INC LDA #\$02	:# OF PLACES STACK IS MOVED
93F1 20 6C 93	826	JSR UPST	:MOVE STACK
93F4 A0 00	827	LDY #\$00	
93F6 20 D3 93	828	JSR AD1	:GET ADDR OF NEXT 2 BYTES
93F9 20 BB 93	829	JSR FBBUF	:STORE DATA IN ADDR IN BUFFER
93FC EE D1 93	830	INC BUF	:ADD 1
93FF D0 03	831	BNE XXINC	:IS BUF=0
9401 EE D2 93	832	INC BUF+1	:YES,INC.HOB
9404 20 C6 93	833	XXINC JSR BUFFB	:STORE DATA

9407 4C 97 93	834	JMP RETURN	:RESTORE X & Y.RETURN
940A	835 *		
940A 00 00	836 KEY2	HEX 0000	
940C 00	837 MOVE	HEX 00	
940D 00 00	838 YX	HEX 0000	
940F	839 *		
940F	840 CALTAB	DFS #72	
94B1	841	END	

\*\*\*\*\* END OF ASSEMBLY

## SAMPLE PROGRAM

```
5  REM *****INITIALIZATION
8  LD = 15: X = 20: Y = 20
10 GOTO 4000
80 REM *****LOOP ROUTINE
100 COLOR= 0: IF CL = 0 THEN COLOR= 15
110 PLOT X,Y
120 COLOR= CL: PLOT X,Y: REM CURRENT COLOR
136 REM *****READ A KEY
150 GET A$
155 REM *****GET FUNCTION NUMBER
158 IF PEEK (768) = 3 THEN 100
160 F = PEEK (779)
170 ON F GOTO 1000,2000,3000,4000,5000,6000
180 GOTO 100
999 REM
1000 REM *****FUN 1 MOVE ONLY
1010 GOSUB 10000: REM GET NEW X,Y
1020 COLOR= LD: PLOT X,Y
1030 X = XN: Y = YN: LD = SCRN( X,Y): GOTO 100
2000 REM *****FUN 2 MOVE &PLOT
2010 GOSUB 10000: REM GET NEW X,Y
2020 COLOR= CL: PLOT X,Y
2030 X = XN: Y = YN: LD = SCRN( X,Y): GOTO 100
3000 REM *****FUN 3 CHANGE COLOR
3010 CL = PEEK (513) - 1: GOTO 100
4000 REM *****FUN 4 CLEAR SCREEN
4010 GR
4015 COLOR= 15
4020 FOR I = 0 TO 39: HLINE 0,39 AT I: NEXT I
4030 HOME : GOTO 100
5000 REM *****FUN 5 SAVE PICTURE
5010 INPUT "PICTURE TO SAVE": P$
5015 ONERR GOTO 5040
5020 PRINT CHR$(4); "BSAVE ": P$; ",A$400,L$400": GOTO 100
5040 HOME : GOTO 100
6000 REM *****FUN 6 LOAD PICTURE
6010 INPUT "PICTURE TO LOAD": P$
6020 ONERR GOTO 6040
6030 PRINT CHR$(4); "BLOAD ": P$; : GOTO 100
6040 HOME : PRINT "FILE NOT FOUND"
6050 FOR I = 1 TO 400: NEXT I: HOME : GOTO 100
9997 REM
9998 REM CALCULATE NEW CURSOR COORDINATES
9999 REM
10000 DX = PEEK (513): DX = DX - 2
10010 DY = PEEK (514): DY = DY - 2
10020 XN = X + DX: YN = Y + DY
10025 IF XN < 0 THEN XN = 39
10030 IF XN > 39 THEN XN = 0
10040 IF YN < 0 THEN YN = 39
10050 IF YN > 39 THEN YN = 0
10060 RETURN
```

## Description

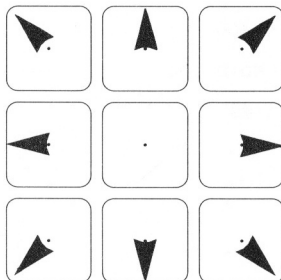
Here is the complete overlay for the **SAMPLE** Application, **SAMPLE**. You can use this overlay with either the **SAMPLE** program or the **BSAMPLE** program and the **BASIC** overlay. Carefully cut out this page and tape it in the lower left corner of your **KEYPORT**. Put the lower left key center on top of Key 46. Boot the KP Monitor Diskette, press **RETURN** to load then monitor. Then enter **RUN SAMPLE**. Try drawing some pictures using this overlay.

## Sample

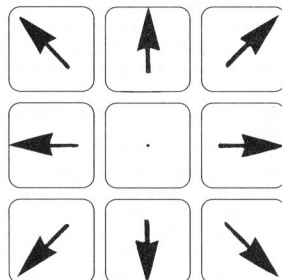
## CURSOR COLOR

BLACK .	RED .	DARK . BLUE	PURPLE .	SAVE . PICTURE
DARK . GREEN	GRAY . 1	BLUE .	LIGHT . BLUE	LOAD . PICTURE
BROWN .	ORANGE .	GRAY . 2	PINK .	CLEAR . SCREEN
LIGHT . GREEN	YELLOW .	AQUA .	WHITE .	

## MOVE CURSOR



## MOVE AND PLOT





The background is a solid blue color with a white grid pattern. The grid consists of squares, and at each intersection of the grid lines, there is a small white diamond shape. This pattern covers the entire page.

# POLYTEL

Computer Products Corp.